Trabajo Fin de Grado
Grado en Ingeniería en Electrónica, Robótica y Mecatrónica

Conversión Analógico-Digital de señales de electroencefalografía

Autor: Jose Araujo González

Tutor: Alfredo Pérez Vega-Leal

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado
Grado en Ingeniería en Electrónica, Robótica y Mecatrónica

Conversión Analógico-Digital de señales de electroencefalografía

Autor:
Jose Araujo González

Tutor:
Alfredo Pérez Vega-Leal
Profesor Titular

Dpto. de Ingeniería Electrónica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo Fin de Grado: Conversión Analógico-Digital de señales de electroencefalografía

Autor: Jose Araujo González
Tutor: Alfredo Pérez Vega-Leal

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Agradecimientos

Es imposible para mí, haber llegado a realizar este último trabajo académico sin la ayuda inestimable de las personas que me rodean, comenzando por mi familia, quienes con esfuerzo y paciencia han hecho de soporte en los cimientos de mi educación. A mi pareja, por enseñarme valores como la constancia y la perseverancia, fundamentales en el camino que he recorrido para llegar a donde estoy. A mis amigos, por compartir los momentos bonitos e igualmente duros que nos ayudaron a madurar, y a todos los profesores y personas que hayan influido en mi educación, por brindar al mundo el conocimiento que nos hará libres.

Jose Araujo González
Sevilla, 2020

Resumen

En este trabajo se llevará a cabo la transformación de señales analógicas de electroencefalografía (previamente amplificadas y filtradas) en señales digitales para su posterior estudio. Por ello el componente principal de análisis será el convertidor analógico-digital *AD7928*, caracterizando los distintos parámetros y comprobar finalmente si es adecuado para dicha finalidad.
Por otro lado, para el desarrollo de este trabajo, se ha hecho uso de distintos dispositivos y comunicaciones para tener como objetivo un compacto sistema de lectura y envío de señales de electroencefalografía hasta un ordenador para su posterior análisis, algo que queda ya fuera del alcance de este documento.
Los componentes principales serán el convertidor analógico-digital *AD7928* conjunto a la placa de desarrollo *Feather HUZZAH ESP8266* bajo programación en el sistema operativo **FreeRTOS**, especializado en sistemas en tiempo real para microcontroladores.

Abstract

In this project will be carried out the conversion of analog electroencephalography signals (previously amplified and filtered) into digital signals for his further study. Therefore, the main component of study will be the analogic-digital converter AD7928 to characterize the differents parameters and checking if it´s suitable for that purpose. Besides that, to develop this project, it has been used various types of devices and communications, being the goal a compact system to read and send electroencephalography signals up to a computer for subsequent analysis, which is out of reach of this paper.
The main components will be the Digital to Analog Converter AD7928 in combination with the development board *Feather HUZZAH ESP8266* programmed with the operating system **FreeRTOS** which is specialized in real-time systems for microcontrollers.

Índice

Notación

CAD	Convertidor analógico-digital
EEG	Electroencefalograma
Hz	Hercio
ms	Milisegundos
SPI	Serial Peripheral Interface
RTOS	Real-Time Operating System
TCP	Transmission Control Protocol
V	Voltios
ROM	Read-only memory
FSR	Full Scale Range
LSB	Least Significant Bit
DNL	Differencial Non-Linearity
INL	Integral Non-Linearity
SNR	Signal-Noise Ratio
THD	Total Harmonic Distorsion
ENOB	Effective Numbers Of Bits
CDA	Convertidor digital-analógico
I2C	Inter-Integrated Circuit

1 Introducción

La ciencia no es sino una perversión de sí misma, a menos que tenga como objetivo final la mejora de la humanidad.

Nikola Tesla

La evolución de la electrónica de consumo ha permitido que tanto usuarios noveles como expertos puedan desarrollar dispositivos baratos y compactos, facilitando que todos los ámbitos en los que son aplicables la electrónica estén más al alcance de la población y se consiga una liberación del saber y de la ingeniería. En este contexto nace la idea de aplicar estos conocimientos en la medicina y así, realizar la captura de señales de electroencefalografía, su conversión al ámbito digital y envío para el posterior análisis, pero de una manera más accesible para la mayoría de la gente.

1.1 Contexto y objetivos

El electroencefalograma es un estudio médico que detecta la actividad cerebral de un individuo por medio de unos electrodos dispuestos sobre la cabeza del mismo. Las células alojadas en el cerebro se comunican a través de impulsos eléctricos y dicha actividad puede ser recogida para el análisis clínico de un paciente o como señal de control para prótesis [1].
Esto es algo que existe desde hace tiempo pero siempre accesible desde el mundo de la medicina, con aparatos costosos y de difícil manejo; sin embargo, desde el grupo *Neurotronics* y el departamento de Ingeniería Electrónica, se intenta crear un sistema completo de lectura, acondicionamiento, conversión y análisis de estas señales, partiendo de ideas como la simplificación y el abaratamiento de costes; pero cumpliendo requisitos mínimos de fiabilidad y seguridad para el usuario.
Así pues, en este trabajo nos centraremos en la tarea de convertir unas señales ya acondicionadas, provenientes de los electrodos anteriormente descritos. Estas señales oscilan entre 1Hz y 100Hz, con un rango de 0V y 5V tras apmlificar y filtar; sin embargo, son señales que contienen mucho ruido *per se* debido al contacto de los electrodos con el cuero cabelludo y de una amplitud muy pequeña -entre μV y mV-. Por lo que, en las etapas posteriores al acondicionamiento -la conversión digital en este caso- se debe intentar que tengan una calidad aceptable, para ello se hará un estudio completo del CAD AD7928 y comprobar si es viable su uso a través de una completa caracterización en el sistema que se presentará.

1.2 Hardware

Para este proyecto, se han dispuesto del microcontrolador Adafruit Feather HUZZAH ESP8266 para poder realizar la comunicación entre el CAD y la computadora que ejecute el análisis. Como ya se comentó anteriormente, el CAD AD7928 ha sido el escogido y estudiado. Por otro lado, de cara a los experimentos para la caracterización del convertidor analógico-digital, se ha hecho uso del CDA MCP4725.

1.2.1 Feather HUZZAH ESP8266

Este microcontrolador de la familia ESP8266, tiene incluido un adaptador *Wi-Fi* para el protocolo TCP a través del cual se enviarán las señales desde el mismo hasta un ordenador.

Tabla 1.1 Características relevantes del Feather Huzzah [2].

Frecuencia de reloj	$80MHz$ o $160MHz$
Memoria Flash	$4MB$
Voltaje	$3.3V$
Comunicaciones	SPI, I2C, UART, Wi-Fi
Modos de bajo consumo	Si

Dicho dispositivo ha sido escogido debido a que posee características mas que satisfactorias para este trabajo por un precio reducido. Además de tener implementado de forma nativa la conexión *Wi-Fi*, algo que se considera primordial en este proyecto por facilitar la portabilidad, además de ser un aislamiento (de manera indirecta) ante la tensión de la red a la que estará conectado el ordenador que realice los análisis.

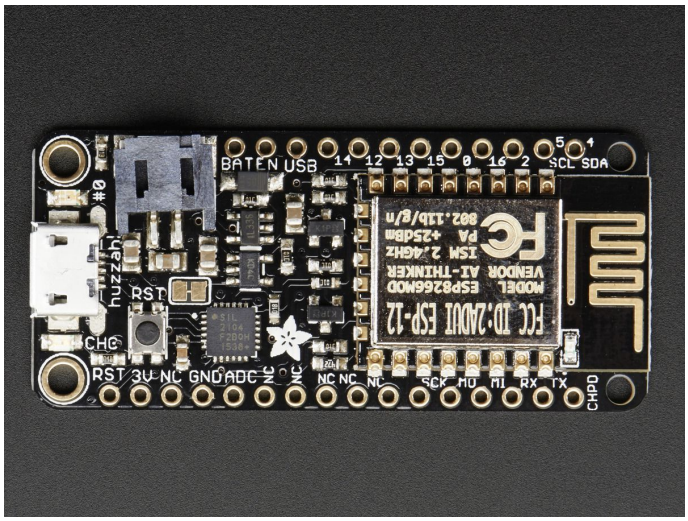

Figura 1.1 Microcontrolador Feather HUZZAH.

1.2.2 Convertidor Analógico-Digital AD7928

Este convertidor escogido de aproximaciones sucesivas, se comunica vía SPI con el microcontrolador y estas son algunas de sus características.

Tabla 1.2 Características del AD7928 [12].

Ratio de muestreo	$1MSPS$
Número de bits	12
Voltaje Referencia (en este caso)	$4{,}095V$
Frecuencia máxima SCLK	$20MHz$
Canales	8

A pesar de haber comentado anteriormente las características específicas de las señales de electroencefalogría, se ha dado como válida -antes de realizar los ensayos oportunos- la elección de este CAD por su cumplimiento teórico en los tiempos de conversión y envío de la información, así como una precisión adecuada para una primera versión.

1.2.3 Convertidor Digital-Analógico MCP 4725

Este CDA de 12 bits, sera partícipe del trabajo como sustituto de un generador de señales a la hora de realizar el montaje experimental. Se encuentra orientado al entorno de desarrollo Arduino y en conjunto con este microcontrolador, componen la generación de todas las señales usadas. Se verán algunos parámetros para entender más adelante si los resultados de los experimentos se pueden considerar válidos por ello:

Tabla 1.3 Características del MCP4725 [3].

Parámetro	Típico	Min/Máx
Tiempo de estabilización	$6\mu s$	-
Número de bits	12	-
Voltaje Referencia	$5V$	$2{,}8V/5V$
Canal	1	-
INL	± 2	$\pm 14{,}5$
DNL	$\pm 0{,}2$	$-0{,}75/\pm 0{,}75$
Error de offset	$0{,}02\,\%FSR$	$0{,}75\,\%FSR$
Error de ganancia	$-0{,}1\,\%FSR$	$-2/2\,\%FSR$

1.3 Software

Para el apartado de software, se ha hecho uso del repositorio de código libre *SuperHouse/esp-open-rtos* [4] para la programación del microcontrolador. Por otro lado, para recibir los datos, su manipulación y visualización de los mismos, Labview ha sido la herramienta escogida, por su sencilla configuración para la comunicación TCP, las gráficas y funciones orientadas en filtrar digitalmente toda la información -algo que queda fuera del alcance de este proyecto-. Para el análisis de los experimentos para caracterizar el convertidor en el sistema, se hará uso de MATLAB.

1.3.1 ESP-OPEN-RTOS

Este desarrollo se basa en un sistema operativo para tiempo real, facilitando la tarea de crear hilos y colas necesarios para el correcto funcionamiento del sistema. Sin embargo en este punto se va a dar unos detalles para instalar este repositorio y programación del microcontrolador, para el futuro en caso de alguien basar su trabajo en éste.
En primer lugar, para poder comunicarnos con la ROM del ESP8266 y realizar distintas configuraciones, hay que descargar la herramienta basada en python **esptool.py** de código libre [5]. Tiene algunas utilidades como: configurar la velocidad del puerto serie en baudios, borrar por completo la memoria flash o leer la dirección MAC del chip *Wi-Fi*. Seguidamente, para poder crear programas y compilarlos en un entorno de tiempo real,

descargar el repositorio **esp-open-rtos** donde hay ejemplos de gran utilidad a la hora de programar desde cero. Llegados a este punto, se da por hecho que para poder hacer uso de este repositorio, es necesario un sistema operativo basado en Linux -Ubuntu 14.04 LTS en este caso-; por esto, detallar unos puntos sobre distintos comandos:

- Para compilar el programa, es necesario que exista en el mismo directorio que en el programa, el archivo *Makefile* y ejecutar en el terminal -de dicho directorio- la orden **make**.
- Cuando la compilación se haya realizado satisfactoriamente, se ejecuta la orden **make flash** y así comenzará la descarga en la memoria flash del microcontrolador el programa compilado.
- En caso de dar fallo a la hora de la descarga, es muy probable que no se tengan los permisos necesarios para escribir a través del puerto USB, para ello se ejecuta **sudo chmod a+rw /dev/ttyUSB0**, siendo /dev/ttyUSB0 la dirección (en este caso) del puerto donde está conectado el microcontrolador.

1.3.2 Labview

Para la comunicación TCP con el microcontrolador, visualización de los datos y futuras características más, Labview se ha tomado como la herramienta idónea. A la hora de realizar las comunicaciones se ha seguido un diagrama de flujo como el siguiente:

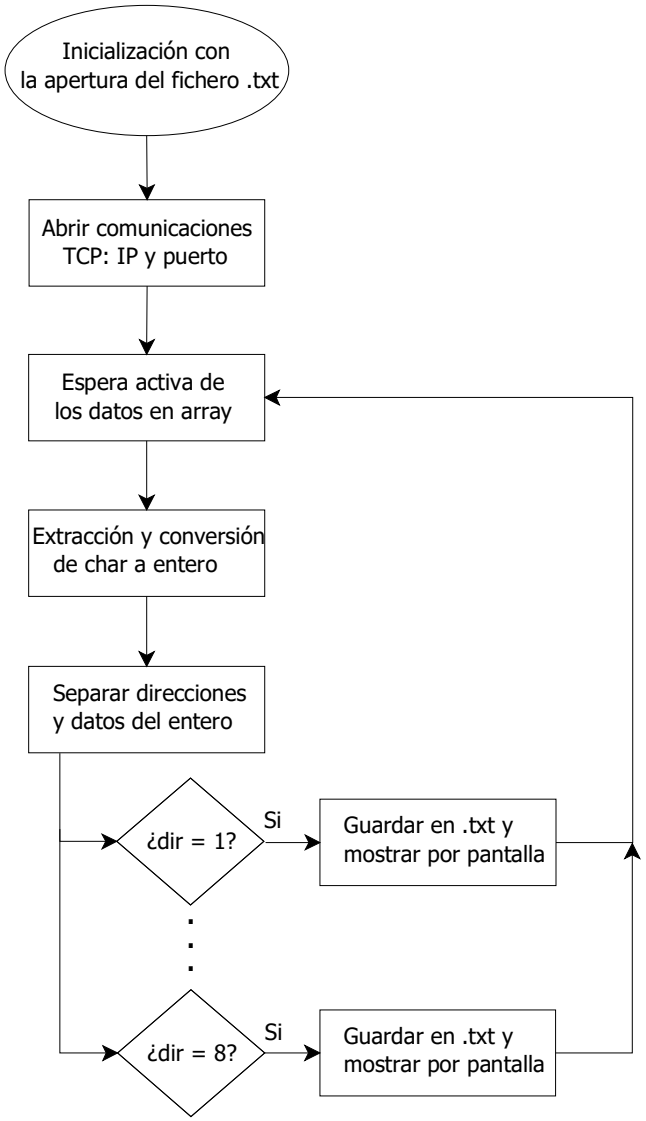

Figura 1.2 Diagrama de flujo de comunicaciones y extracción de datos.

Donde se hace especial hincapié en la correcta conversión de los datos desde *char* a entero sin signo, pues el hecho de ir un valor tras otro en un *array* provocaba ciertas dificultades, pero desde el punto de vista del envío de la información era lo más efectivo. Por otro lado también se intentó tener en mente que este proyecto es parte de otro mayor, pues estos datos se disponían para una filtración digital y su uso para el análisis de ondas de electroencefalografía.
También se adjunta un detalle sobre el montaje gráfico de Labview, facilitando su visualización como banco de pruebas y detección de errores, de la manera más sencilla posible.

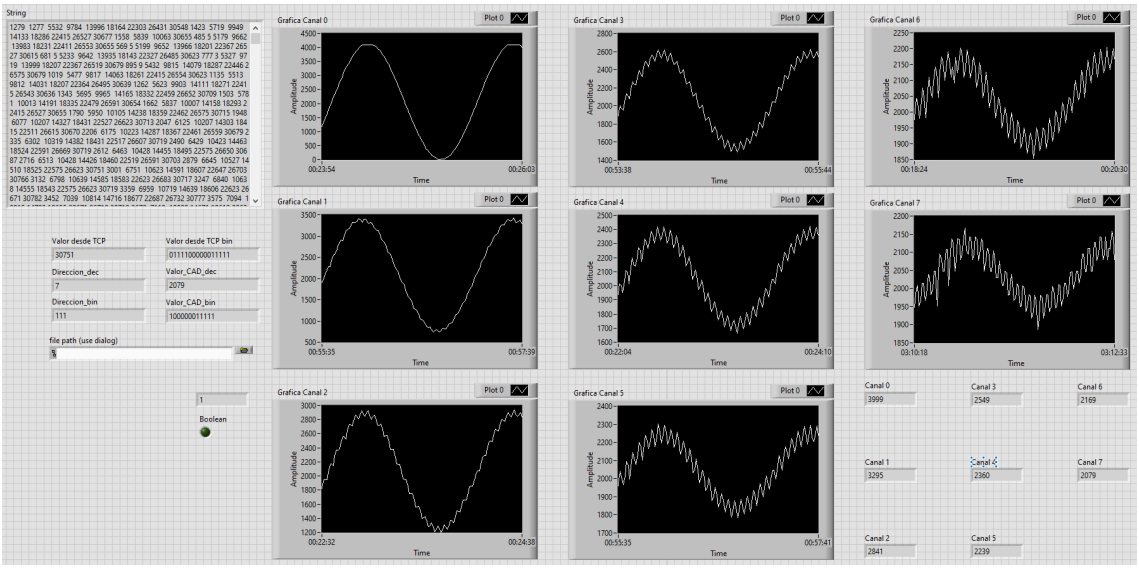

Figura 1.3 Pantalla de Labview para visualización de los datos.

1.3.3 MATLAB

Para el análisis de la información recibida y guardada en un archivo de texto plano por parte de Labview, se ha hecho uso de MATLAB como herramienta fundamental, gracias a la potencia de las funciones que tiene a la vez que su sencillez para la manipulación de datos. A pesar de que existe la posibilidad de crear otra comunicación TCP entre ambos programas, se propuso que no fuera así para evitar mayores retrasos en la señal; además de que es inmediata la creación y carga del archivo en texto plano desde Labview hacia MATLAB. Más adelante se verá en el apartado de experimentos, los distintos programas y funciones usadas para el análisis.

2 Conversión analógico-digital

Es difícil decir que es imposible, porque el sueño de ayer es la esperanza de hoy y la realidad de mañana.

ROBERT H.GODDARD

En este primer capítulo, se va a abordar el eje central del trabajo aquí expuesto, la conversión digital de señales analógicas. Se comenzará por presentar los distintos tipos de convertidores existentes,así como, los parámetros con mayor relevancia de un convertidor para después aterrizar en el dispositivo de estudio, el AD7928.

2.1 Tipos de convertidores analógico-digitales

De entre los distintos tipos de arquitectura existentes en el mercado, se van a exponer los más populares, para así entender qué tipo de convertidor se trata el AD7928 y el por qué de su elección.

2.1.1 Convertidor Flash

El convertidor Flash tiene la capacidad de realizar cualquier conversión en un solo ciclo de medición, esto se debe a que hace uso de $2^n - 1$ comparadores para medir con una resolución de **n** bits, los cuales trabajan en paralelo sobre la tensión de entrada, tras esto basta con decodificar los datos ya digitales.

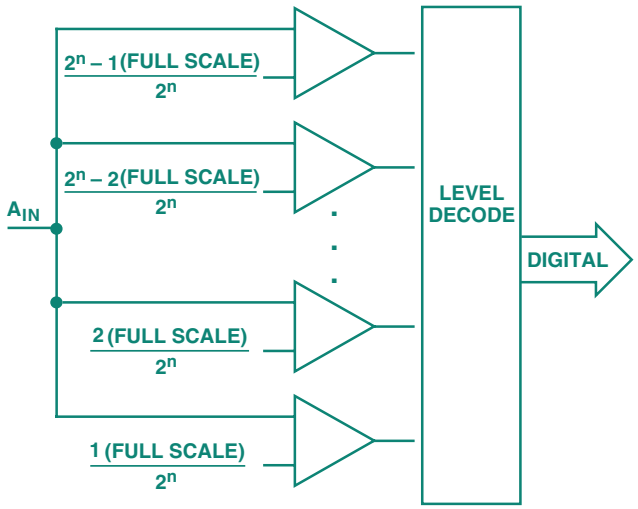

Figura 2.1 Esquema básico de la estructura del convertidor Flash [6].

Es por su tipo de estructura, que los convertidores Flash son usados normalmente para conversiones muy rápidas, a costa de la dificultad de tener muchos comparadores encapsulados y pérdidas de linealidad, por ello suelen tener números reducidos resolución de bits. En el caso de las señales de electroencefalografía, no requieren una rápida conversión -las más rápidas son de unos 100 Hz-, pero si se requiere una mayor cantidad de resolución debido a su pequeña amplitud y cantidad de ruido inherente.

2.1.2 Convertidor Sigma-Delta

Desde un punto de vista sencillo, estos convertidores están formados por un integrador, un comparador y un CDA.

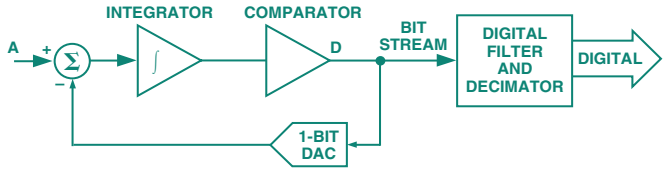

Figura 2.2 Diagrama por bloques del convertidor Sigma-Delta.

A la señal analógica se le resta el resultado del CDA, que pasa a ser integrada, pasa por un comparador y dicha salida vuelve al CDA, suponiendo que éste es de un solo bit, tendremos una ristra de '1'y '0', donde la cantidad de valores de nivel alto es proporcional al valor analógico. Finalmente dichos valores son filtrados y pasados a decimal.
A pesar de tener la característica de pasar ruido de bajas frecuencias a otras más altas como punto favorable, no es demasiado usado en sistemas de señales multiplexadas, debido a la latencia y el sobremuestreo, pues habría que agregar tiempos muertos entre medidas. Esto se soluciona en sistemas más complejos que se salen del fin de este trabajo, como el abaratamiento de costes.

2.1.3 Convertidor de aproximaciones sucesivas

En último lugar, el convertidor de aproximaciones sucesivas, tiene una arquitectura formada por un *sample and hold*, un CDA y un solo comparador:

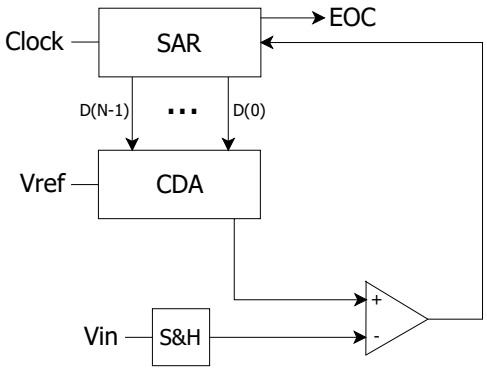

Figura 2.3 Arquitectura convertidor SAR.

El proceso que sigue para realizar una conversión completa es a base de comparaciones, como si de pesos se tratase: en primer lugar compara si el valor de entrada es mayor o menor que la mitad del fondo de escala, seguidamente si es mayor o menor que 1/4 del fondo de escala y así de manera sucesiva, tantas veces como bits tenga el convertidor.

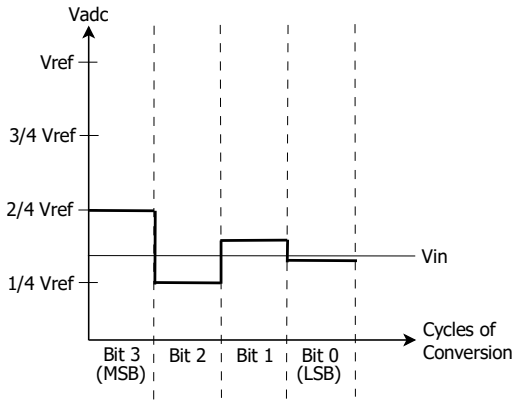

Figura 2.4 Aproximaciones de un convertidor SAR de cuatro bits.

Este tipo de convertidor ha sido el elegido para el trabajo que se presenta, debido a algunas ventajas y características que lo hacen adecuado, como son las siguientes:

- Puesto que necesita **n** ciclos para la conversión completa, es adecuado para sistemas de frecuencias bajas.
- Como su arquitectura es sencilla, los componentes que lo conforman pueden ser de calidad y tener una alta resolución manteniendo un precio bajo.
- Las conversiones pueden comenzar a realizarse cuando se desee, por ese motivo es también adecuado para sistemas con señales no periódicas.
- Es común usar un multiplexador de señales a la entrada de un SAR, como sustituto menos caro a los demás convertidores

2.2 Parámetros y características

2.2.1 Fondo de escala

Se define el fondo de escala (FSR) de un convertidor como los extremos de la señal analógica a la entrada [7]. En este caso al tratarse de un convertidor unipolar, va desde los 0V hasta los 4.095V; se escogió este valor para que, las conversiones de analógico a digital, tuvieran valores sencillos, como se verá más adelante en el **Bit menos significativo** . Por ello $FSR = 4.095V - 0V = 4.095V$.

2.2.2 Bit menos significativo

El bit menos significativo (LSB) es el mínimo valor que el CAD puede convertir a digital o dicho de otra manera, el menor incremento a la entrada que efectúa un cambio a la salida del convertidor.
Por su propia definición, la manera de calcular el LSB es dividiendo el fondo de escala entre la cantidad de valores posibles que tiene el convertidor, en este caso $LSB = \frac{FSR}{2^n - 1}$ siendo **n** el número de bits: $LSB = \frac{4.095}{4095} =$ 0,001.

2.2.3 Curva característica

La curva característica de un CAD o función de transferencia, es una gráfica donde se presenta el voltaje de entrada frente al código binario que genera el CAD como salida. No es una gráfica continua, sino discreta de 2^n puntos -4096 en este caso- [8].

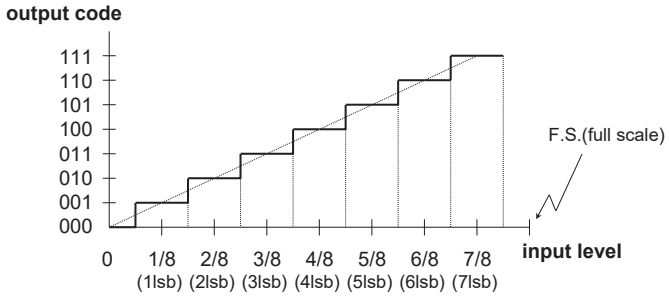

Figura 2.5 Curva característica ejemplo de un CAD de 3 bits.

Sin embargo, debido a limitaciones en el proceso de fabricación, los convertidores reales no tendrán esta función de transferencia exacta -como se verá en el punto de caracterización del CAD-. Estas desviaciones se pueden dividir en dos: error en el *offset* y error en la ganancia.

2.2.4 Error de offset

Este error se ve como el desplazamiento del eje horizontal que tiene la función de transferencia real frente a la ideal. Si el error es positivo, implica que en tensiones altas saturará antes de que la señal de entrada llegue al máximo, y al contrario, si el error de offset es negativo, para pequeñas tensiones a la entrada, se obtendrá un 0 a la salida del convertidor analógico-digital.

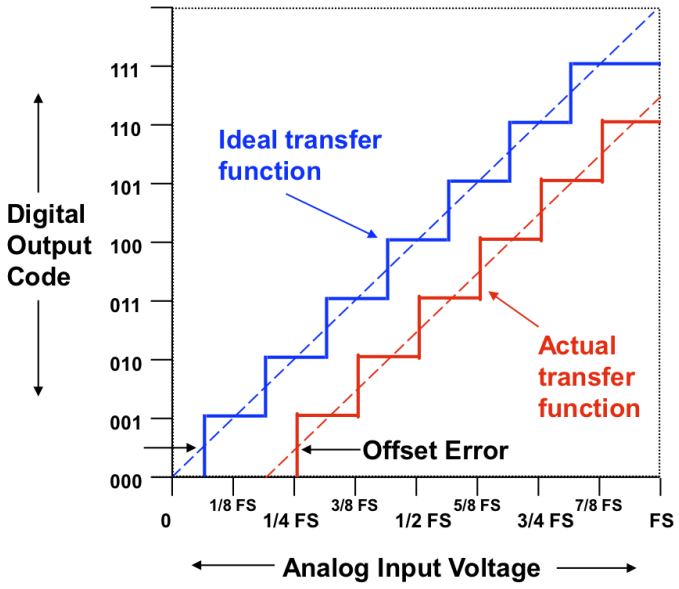

Figura 2.6 Error de offset sobre curva característica.

2.2.5 Error de ganancia

En este otro caso, la desviación de la gráfica aumenta conforme el valor de la señal de entrada es mayor. El comportamiento que ello conlleva es que, ante un error en la ganancia positiva, la salida saturará antes de que lo haga la entrada al convertidor; en caso contrario, si la ganancia es negativa, no llegará al valor máximo de salida ante el valor máximo de la entrada. Normalmente se debe a un error en la referencia de voltaje del convertidor.

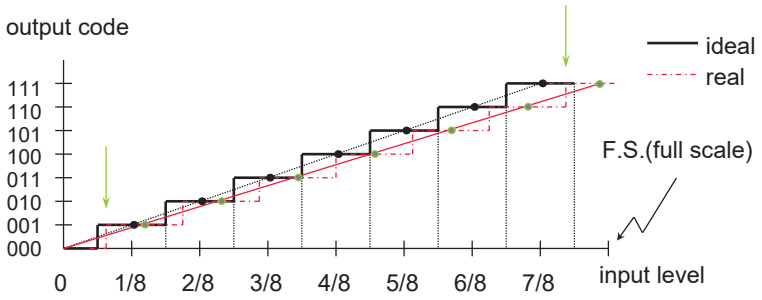

Figura 2.7 Error de ganancia sobre curva característica.

2.2.6 No linealidad diferencial

La no linealidad diferencial son los desajustes entre transiciones de un valor digital del CAD real a otro -mirando en la curva característica- con respecto al ideal. Si el DNL es menor que $-1LSB$, tendremos el denominado problema de *missing codes*; es decir, valores a la entrada que no tendrán un valor a la salida del convertidor.Se puede calcular de la siguiente manera:

$$DNL_i = \frac{ancho\_real_i - ancho\_ideal_i}{ancho\_ideal_i}, i = 0...2^n - 1$$

$$DNL = Sign(DNL_i) \cdot Max(|DNL_i|), i = 0...2^n - 1$$

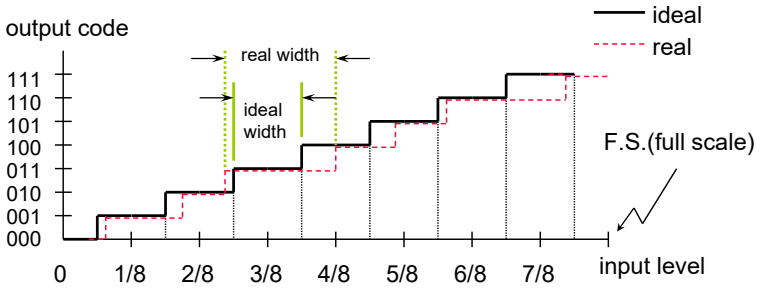

Figura 2.8 No linealidad diferencial entre curvas real e ideal.

2.2.7 No linealidad integral

La no linealidad integral se define como la máxima desviación entre los puntos de transición de la curva característica ideal y real.

$$INL_i = INL_{i-1} + DNL_i = \sum_{j=1}^{i} DNL_i, i = 0...2^n - 1$$

$$INL = Sign(INL_i) \cdot Max(|INL_i|), i = 0...2^n - 1$$

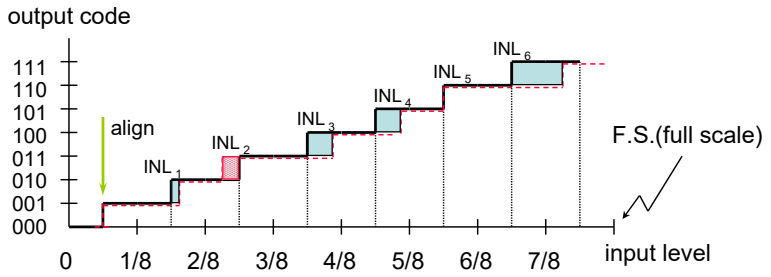

Figura 2.9 No linealidad integral entre curvas real e ideal.

2.2.8 Ruido de cuantización

Este ruido o incertidumbre proviene de la propia naturaleza del convertidor, debido que a la hora de cuantificar el valor exacto de la señal de entrada, esta posee una precisión infinita -en matemáticas- frente a la precisión finita del convertidor, lo cual está íntimamente relacionado con el bit menos significativo.
Este error se puede suponer de distribución uniforme en su amplitud y de ruido blanco en toda la banda de Nyquist y se mide como la diferencia entre la señal de entrada y la dada por el convertidor; teniendo de manera general una forma de dientes de sierra [9].
Así pues, al tener forma de dientes de sierra, se encontrará en el rango:

$$-\frac{LSB}{2} < e_q(n) < \frac{LSB}{2}$$

Y, además, conociendo que la potencia de una señal en forma de dientes de sierra es:

$$\frac{Amplitud^2}{12} = \frac{LSB^2}{12} = \frac{0.00122^2}{12} = 1{,}2403 \cdot 10^{-7}$$

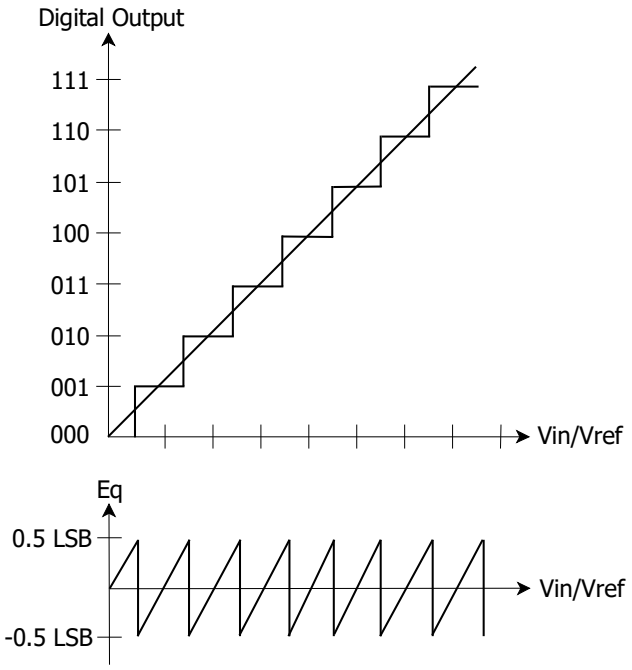

Figura 2.10 Error de cuantización.

2.2.9 Relación señal/ruido

Se define como la proporción de la potencia de la señal de entrada sobre la potencia de ruido y expresado en decibelios. Así pues, el cálculo de este ratio según su propia definición

$$SNR(dB) = 10 \cdot log\left(\frac{\sigma_{signal}}{\sigma_{noise}}\right)$$

En este caso, el SNR no incluye la distorsión provocada por los armónicos pero sí el ruido de cuantización -además de otros como el ruido térmico, shot o flicker [10]-.
Sin embargo, para el caso ideal, la relación señal/ruido depende de la cantidad de bits del convertidor, siendo el cálculo teórico de la siguiente manera:

$$SNR(dB) = 6.02 \cdot n + 1.76 = 6.02 \cdot 12 + 1.76 = 74dB$$

2.2.10 Distorsión Armónica

La distorsión armónica (THD) relaciona la suma de todos las amplitudes de los armónicas de la señal frente a la amplitud de la misma:

$$THD(dB) = 20 \cdot log\left(\frac{\sum_{i=2}^{n} V_{fn}^2}{V_{f1}^2}\right)^{1/2}$$

Estos armónicos se producen de manera inherente al convertidor debido a su naturaleza no lineal, precisamente por ello, una manera de reducirlos es haciendo parecer su curva característica a una línea lo máximo posible. Como suele ser común, tan solo los primeros armónicos son los que se tienen en cuenta a la hora de realizar el cálculo del THD.

2.2.11 Número efectivo de bits

Este número (ENOB) resulta ser la cantidad de bits "reales" que posee el convertidor; en otras palabras, a pesar de haber sido fabricado con un número de bits predeterminado -12 en nuestro caso-, puede que debido a la falta de linealidad del convertidor y el ruido del circuito en el que se encuentre, el ENOB no coincida con la cantidad de bits que tiene el CAD.
Se encuentra íntimamente relacionado con el SNR, y por ello, posee la misma fórmula despejando **n**:

$$ENOB = \frac{SNR - 1{,}76}{6{,}02}$$

Debido a esa relación con el SNR, generalmente el ENOB disminuye en altas frecuencias

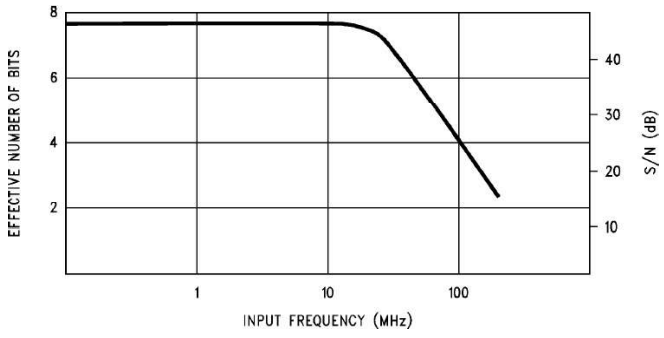

Figura 2.11 Función del ENOB frente a la frencuencia.

3 Comunicaciones

Hacerte invencible significa conocerte a ti mismo.

El arte de la guerra, Sun Tzu

Las comunicaciones no son el estudio principal de este proyecto, pero sí suponen un punto vital en el funcionamiento del sistema al completo, debido a que nos proporcionan portabilidad y aislamiento físico de la red eléctrica. En este caso se han hecho uso de los protocolos SPI y TCP.

3.1 Bus SPI

El bus SPI es un sistema de comunicaciones con una arquitectura **maestro-esclavo**, donde el dispositivo maestro -el microcontrolador en este caso- comienza la conexión con el esclavo -el CAD-, siendo además una comunicaciones *full-duplex* [11], pues ambos dispositivos leen datos y envían al mismo tiempo. El esquema básico de conexión es el siguiente:

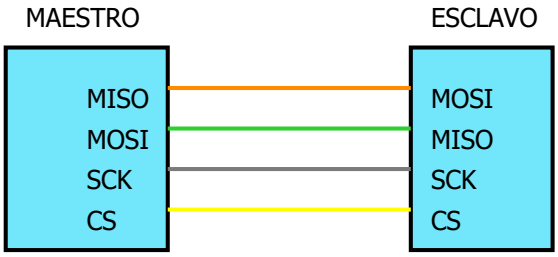

Figura 3.1 Esquema bus SPI.

- **MISO** (Master-In, Slave-Out): línea donde la comunicación transcurre del esclavo al maestro.
- **MOSI** (Master-Out, Slave-In): línea donde la comunicación transcurre del maestro al esclavo.
- **SCK**: línea en la que el maestro comparte la señal de reloj.
- **CS** (Chip-Select): bit de activación -a nivel bajo- de la comunicación con el esclavo.

Dicha comunicación es síncrona; es decir, el microcontrolador aporta una señal de reloj al CAD, para que éste sepa la velocidad de lectura y escritura del bus. Debido a que este protocolo está concebido para varios esclavos, se hace uso de una línea adicional para seleccionar qué dispositivo va a realizar la comunicación con el maestro.

En la programación del microcontrolador, se tuvieron que tener en cuenta además otros parámetros de conexión entre el maestro y el esclavo a la hora de inicializar la función *spi_init* [4], estos fueron los argumentos:

- Elección del bus SPI, en este caso fue el Bus 1 del *Feather Huzzah*
- Polaridad y fase del reloj, $CPOL = 0$, implica que la señal de reloj se encuentra a nivel bajo cuando no está en funcionamiento y ofrece pulsos a nivel alto, donde realizará la lectura [12] y $CPOL = 0$, que hará coincidir los flancos de subida, cuando el valor del bit se encuentre aproximadamente por la mitad del ciclo de reloj.
- Frecuencia de la señal de reloj, se dispuso a lo máximo disponible desde el comienzo para prevenir retrasos: $20MHz$.
- *Big endian*: es decir, que la palabra de 16 bits que se envía, comienza por el bit más significativo.
- *Minimal pins*: un valor booleano -**true** para activar esta opción- que nos permite usar la señal de CS en cualquier salida digital que se desee, activada para poder tener más control del programa.

3.1.1 Registro de control

El registro de control que se encuentra en el convertidor tiene un tamaño de 12 bits que se leen a lo largo de 16 ciclos de la señal de reloj **SCK** [12], teniendo el registro la siguiente composición:

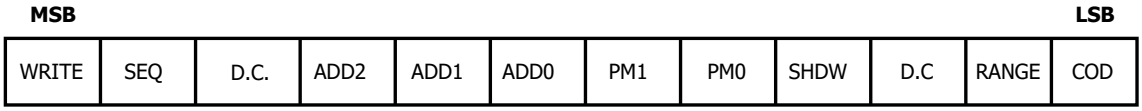

Figura 3.2 Registro de control del CAD.

A continuación se muestra una tabla con los bits de mayor relevancia para el trabajo.

Tabla 3.1 Tabla funciones de los bits del registro.

Bit	Mnemónico	Comentario
11	WRITE	Este bit estará siempre a 1 para escribir en el registro la dirección del próximo canal a convertir.
8 al 6	ADD2 al ADD0	Estos tres bits son en los que se escribirá la dirección del siguiente canal para la conversión a digital, numerándose de 0 a 7.
5 al 4	PM1 al PM0	Ambos bits estarán siempre a 1, para un funcionamiento normal del convertidor.
0	CODING	Este otro bit también estará siempre a 1 para recibir los datos en una misma ristra de 16 bits.

3.1.2 Resultado de la conversión y diagrama temporal

Con cada petición del maestro hacia el esclavo, éste devuelve al mismo tiempo la conversión del último canal descrito por el maestro en una ristra de 16 bits de la siguiente manera:

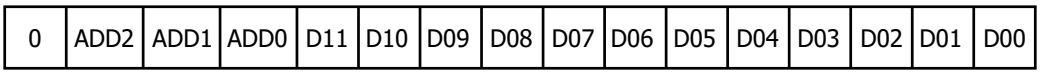

Figura 3.3 Resultado de cada conversión.

Así pues, el bit más significativo estará siempre a 0 (dato sin relevancia), los siguientes tres bits indican el

canal cuyo valor se ha convertido en los bits del **D11** al **D0**.
A la hora de la programación del *Feather Huzzah*, se decidió que el análisis de la dirección del valor en cada lectura del convertidor, estuviera a cargo del sistema destino debido a dos principales motivos:

- Dejar el microcontrolador libre de funciones que pueden realizar otros sistemas, pues desde el inicio se comenzó a programar sin conocer si los tiempos se iban a cumplir adecuadamente; pues se prefería que el microcontrolador estuviera en espera activa constantemente, como prevención.
- Poder discernir qué canal estoy enviando; al comienzo se estudiaron varias posibilidades para facilitar la recepción de la información al sistema destino, pero siempre iba a ser más sencillo y efectivo tener codificados en 3 bits al comienzo de la ristra de los 12, la dirección del canal recién leído.

Por otro lado, para la correcta programación del microcontrolador y recogida de datos, se ha seguido el diagrama temporal dado por el fabricante.

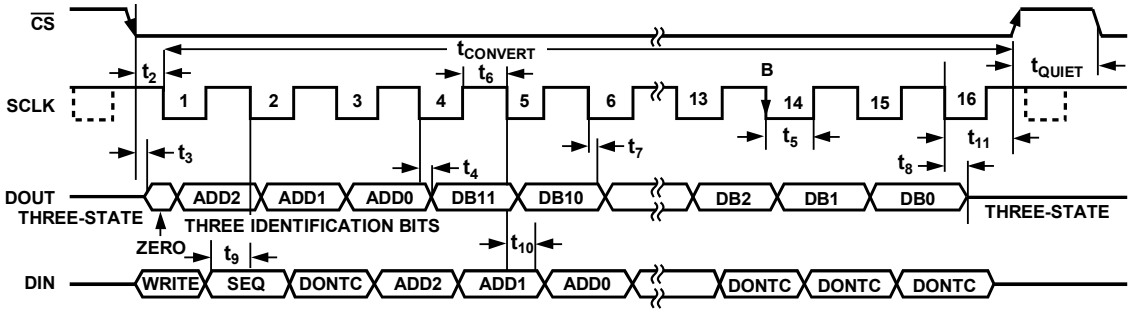

Figura 3.4 Diagrama temporal bus SPI.

3.2 Protocolo TCP

El protocolo de control de transmisión es un servicio orientado a una conexión fiable, *full-duplex* y de punto a punto [13]. Siendo el microcontrolador el proceso servidor y la computadora que recibe los datos, el proceso cliente. La conexión por ello, se realiza a través del puerto 23 (perteneciente al protocolo TELNET pero que no se usa al mismo tiempo que el TCP) hacia cualquier IP privada que quiera escuchar los datos, para simplificar el problema.
Por la propia construcción de la trama TCP, los datos deben ser enviados en tipo **char**, algo que hay que tener en cuenta a la hora de estudiar los datos enviados y la reconstrucción de los mismos. Para el envío y recepción de las tramas con los datos incorporados, tanto en FreeRTOS como en Labview existen funciones específicas que facilitan la implementación de este protocolo y creación de las tramas TCP:

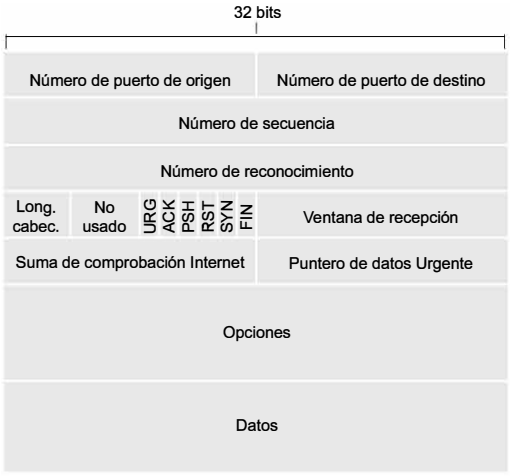

Figura 3.5 Segmento protocolo TCP.

Como se puede comprobar en la figura 3.5 existen muchos campos de información necesaria para el correcto funcionamiento del protocolo que no es necesario construir; de esta manera y por simplificación, en el campo **Datos** se introduce la información de 16 bits sin manipular.

Para tener una idea sobre los retrasos en las comunicaciones, que no significa un retraso en la captación de las señales, se realizó *ping* al ESP8266. Los tiempos sugeridos en esta comprobación no son demasiados pequeños, pero de momento, aceptables.

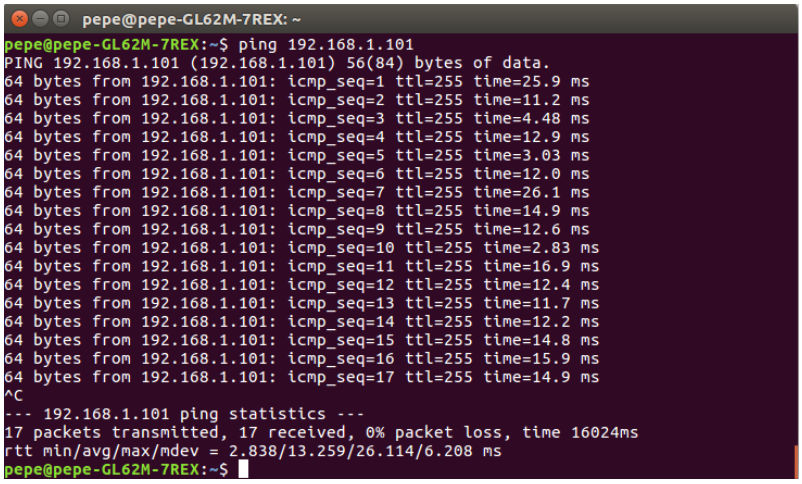

Figura 3.6 Comando *ping* al microcontrolador.

Con respecto al esquema de conexión, se hizo uso de un *router* con tecnología *Wi-Fi* que nos permitiera mayor comodidad y potencialmente, mandar esos datos por internet en versiones posteriores de este proyecto; de ahí a que se desechara la primera idea de una red *ah hoc* punto a punto, por la portabilidad y envío a través de la red -lo cual requiere uso de ciberseguridad, pues los valores médicos generados en un electroencefalograma se encuentran bajo protección de datos-.

4 Sistema montado al completo

Aprendí que el coraje no era la ausencia de miedo, sino el triunfo sobre él.

NELSON MANDELA

Una vez presentado cada parte individual de este proyecto, en este apartado se hará una visión mas global, tanto a nivel de conexión hardware, como funcionamiento del software programado.
Haciendo el seguimiento del flujo de los datos, el diagrama de cómo viaja la información sería el siguiente:

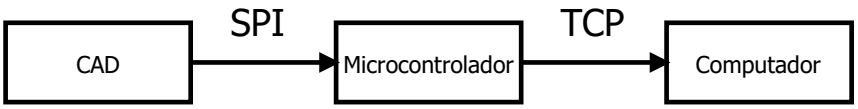

Figura 4.1 Diagrama básico del flujo de información.

Y por otro lado, la conexión y cableado de los distintos dispositivos:

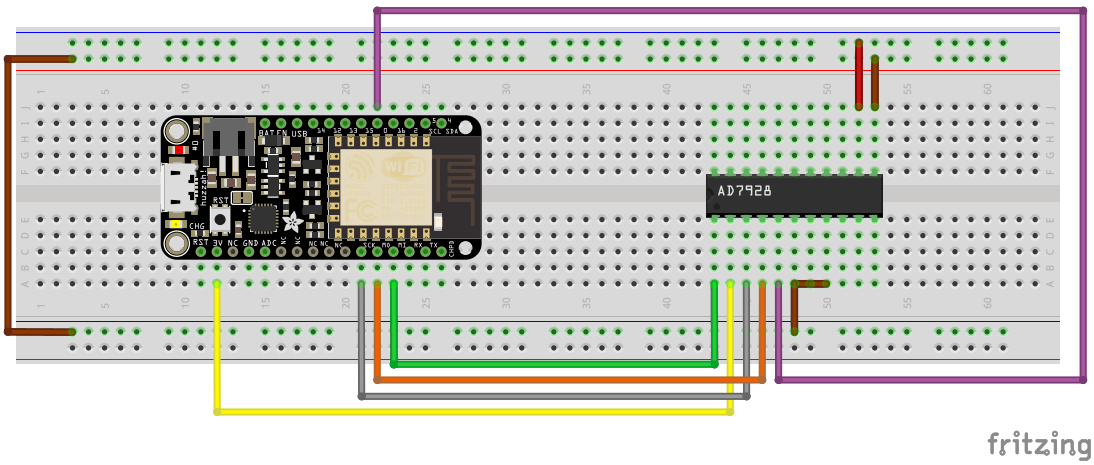

Figura 4.2 Esquema físico de conexión.

A la hora de programar el microcontrolador, se ha seguido el siguente diagrama de flujo:

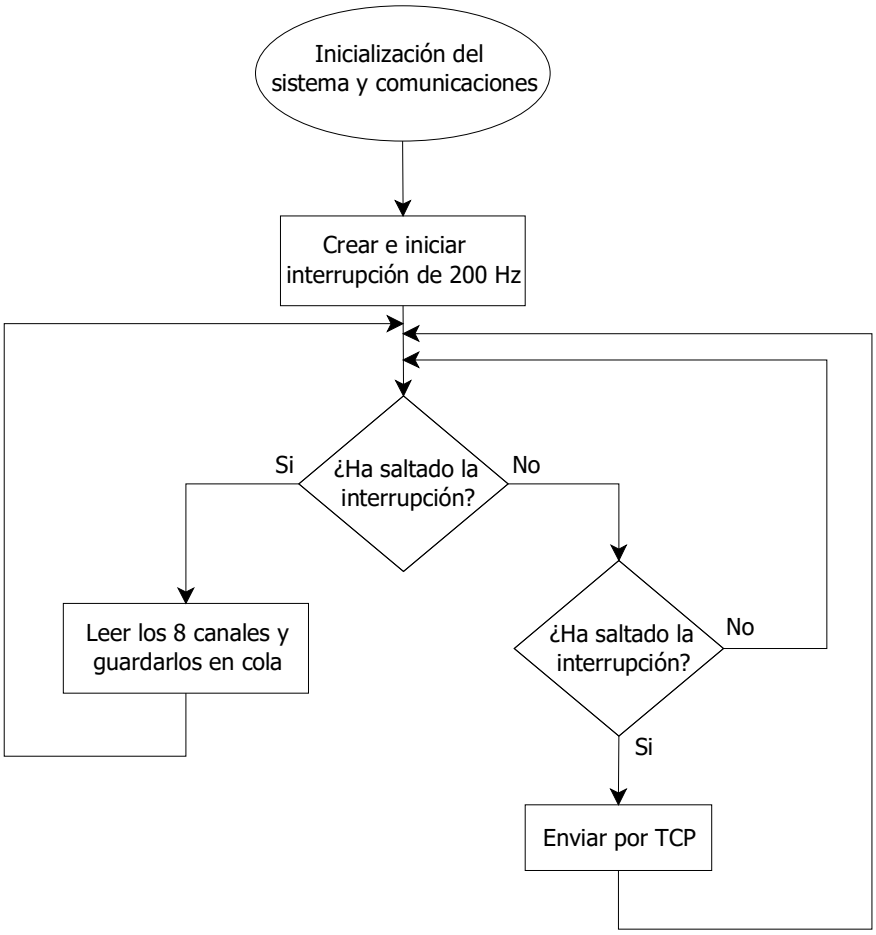

Figura 4.3 Diagrama de flujo de la programación.

5 Experimentos y caracterización del CAD

Nuestra mayor debilidad radica en renunciar. La forma más segura de tener éxito es siempre intentarlo una vez más.

THOMAS A. EDISON

A lo largo de este capítulo, se irán abordando las distintas pruebas y experimentos que se han hecho tanto al sistema al completo, como al convertidor de una manera más individual.
Los requisitos para que el sistema al completo se considerara válido era el mero cumplimiento de los tiempos y la frecuencias a la hora de captación de las señales del CAD y su posible extrapolación a una mayor cantidad de canales, debido a que, en un electroencefalograma, se hará uso de unos 21 electrodos, cada uno ocupará un canal y teniendo en cuenta que el convertidor AD7982 posee 8 canales, serán necesarios dos convertidores más.
Debido a que la parte experimental de este trabajo se ha realizado en un momento donde era imposible acceder a equipos de laboratorio, se pensó en las posibilidades a la hora de realizar los experimentos, donde la primera opción era un circuito RC de carga y descarga que, alimentado con un *duty cycle* del microcontrolador, se podía obtener a la salida una señal periódica; sin embargo, esto iba a resultar más laborioso a la hora del análisis dinámico, pues habría que haber estudiado muy bien la señal desde el punto de vista del espectro en frecuencia para averiguar qué frecuencias corresponden a la señal y cuáles no. De este modo, se encontró -bajo la premisa del abaratamiento de costes-, el convertidor digital-analógico MCP4725 orientado para microcontroladores con comunicación I2C, controlado por un Arduino Mega 2560, como se verá en experimentos posteriores, un circuito RC a modo de filtrado de señales de alta frecuencia compuestos por un potenciómetro y un condensador. Todo ello prototipado en placas de desarrollo con cables estilo dupont para el conexiado, se adjunta imagen del montaje de los experimentos:

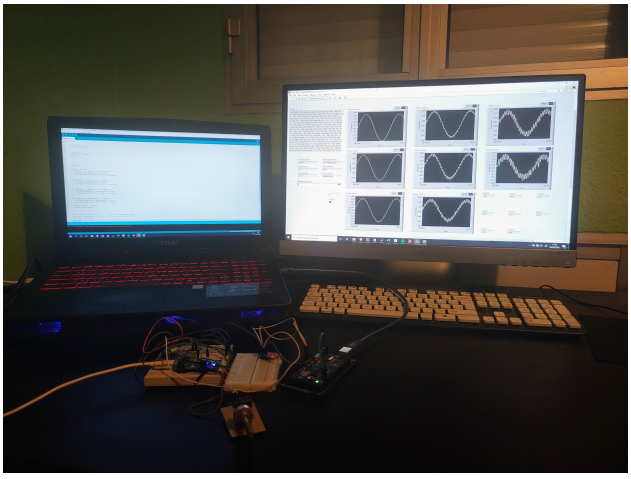

Figura 5.1 Montaje del banco de pruebas.

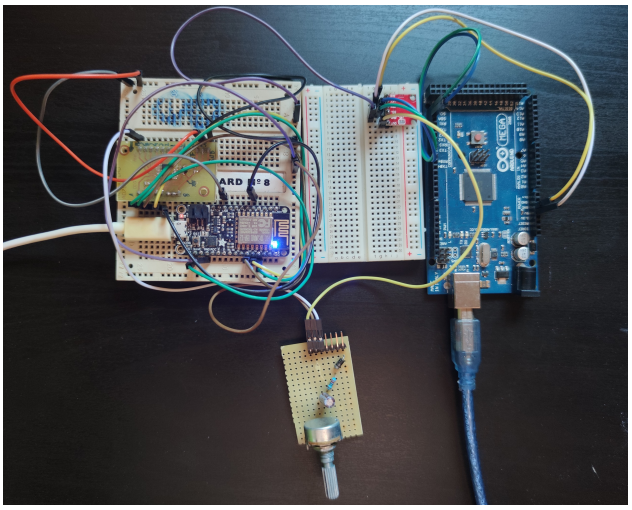

Figura 5.2 Detalle del circuito de pruebas.

5.1 Experimentos temporales

Como requisito fundamental en este primer punto, se propuso que la captación y envío por TCP de los 8 canales fuera más rapido que la frecuencia Nyquist -recordemos que en este caso son 200 Hz- o al menos la captación de las señales y su guardado en una cola.
Tras haber programado el microcontrolador para que la conversión y las comunicaciones SPI fueran lo más rápidas posible, quedaba observar el tiempo real de cada conversión, la interrupción al completo de los 8 canales, así como el envío de las tramas, los resultados fueron los siguientes:

- La interrupción de lectura de los 8 canales del CAD tiene una duración de $60us$.
- El envío de un solo canal se encuentra en el orden de $24us \sim 30us$, con lo que, en condiciones normales de funcionamiento nos encontramos ante un tiempo total de $8 \cdot 24us \sim 8 \cdot 30us = 192us \sim 240us$. Sin embargo, se observaron en algunas ocasiones tiempos más largos y un sólo canal llegó a tardar hasta $130us$

Estas pruebas se intentaron en primer lugar, con funciones propias del sistema operativo para tiempo real que miden instantes de tiempo en microsegundos, pero no eran pruebas precisas debido a que, dentro de la interrupción y en algunas funciones TCP, dichas funciones podían no funcionar correctamente y los resultados eran muy dispares. Finalmente se optó por activar una salida digital, observarla mediante un osciloscopio y contar los microsegundos que ha estado activa la señal.

5.2 Caracterización del convertidor

Para la comprobación de la bondad del dispositivo, así como sus características embebido ya en el sistema; se ha organizado una serie de experimentos, tanto estadísticos como paramétricos del convertidor. Como se detallará más adelante, la mayoría de ellos se han hecho bajo circunstancias que imposibilitan una buena exactitud del resultado de dichos experimentos.

5.2.1 Distribución de la medida a entrada constante

Para la descripción básica, en primer lugar, se realizaron del orden de 16.000 muestras con una entrada constante de tensión de $3{,}3V$; con el fin de entender el error sistemático del convertidor y su fiabilidad. Los resultados son los siguientes:

Tabla 5.1 Media y desviación típica (unidades de salida del convertidor).

Media	3347,802068
Desviación típica	2,021147543

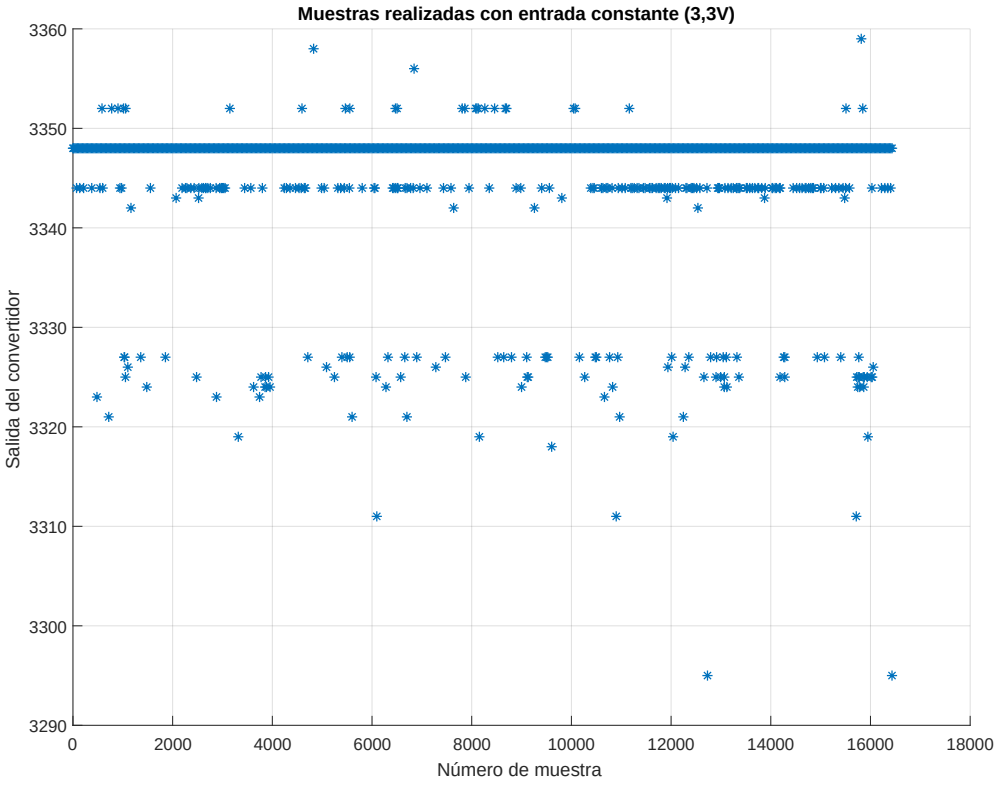

Figura 5.3 Muestras de la salida del convertidor.

Tabla 5.2 Distribución en porcentaje.

3295	0,0122 %
3311	0,0182 %
3318	0,0061 %
3319	0,0243 %
3321	0,0304 %
3323	0,0243 %
3324	0,0791 %
3325	0,1885 %
3326	0,0365 %
3327	0,2311 %
3342	0,0365 %
3343	0,0243 %
3344	1,1861 %
3348	97,926 %
3352	0,1582 %
3356	0,0061 %
3358	0,0061 %
3359	0,0061 %

De esta manera, se observa que tras 16440 medidas realizadas a lo largo del tiempo, la mayoría se encuentran en el valor de 3348. Para comprender mejor el alcance estadístico de la dispersión en la medida, se ha propuesto la tabla 5.2, donde se observa en porcentaje como se distribuye los valores de salida del convertidor; destacando el 97 %, que no implica necesariamente una buena exactitud del CAD en la medida. Por otro lado, se propuso calcular, suponiendo que las tomas de muestras siguen una distribución normal, ¿cuántas medidas son necesarias para tener un error menor a un LSB con una probabilidad del 90 %?. Lo cual se calculó de la siguiente manera:

$$P = 1 - 2\alpha = 0{,}9 \rightarrow \alpha = \frac{1-0{,}9}{2} = 0{,}05 \rightarrow 1 - \alpha = 0{,}95 \rightarrow (tablas)1{,}64 \sim 1{,}65 \rightarrow$$

$$\rightarrow \frac{\varepsilon}{\sigma} \cdot \sqrt{n} = 1{,}64 \rightarrow \frac{1(LSB)}{2{,}0211(CAD)} \cdot \sqrt{n} = 1{,}64 \rightarrow n = 10{,}986 \rightarrow n = 11 medidas$$

5.2.2 Parámetros estáticos del convertidor

Para poder caracterizar los parámetros que se describieron anteriormente, se tuvo que hacer uso del convertidor digital-analógico MCP4725 de 12 bits para realizar todos los siguientes experimentos. A pesar de que fuera deseable haber dispuesto de un generador de señales con mayor resolución, las pruebas se realizaron con la mayor meticulosidad posible y tomando como referencia una buena fiabilidad de dicho convertidor digital-analógico.

- **Curva característica**: este experimento se realizó creando una rampa de valores desde $0V$ hasta $4.095V$, dicha rampa se compone de saltos de 1 LSB debido a que ambos convertidores tienen la misma resolución. Los datos se fueron tomando aproximadamente cada segundo y se realizó una media la recogida de valores en cada escalón. La gráfica de la curva característica es la siguiente:

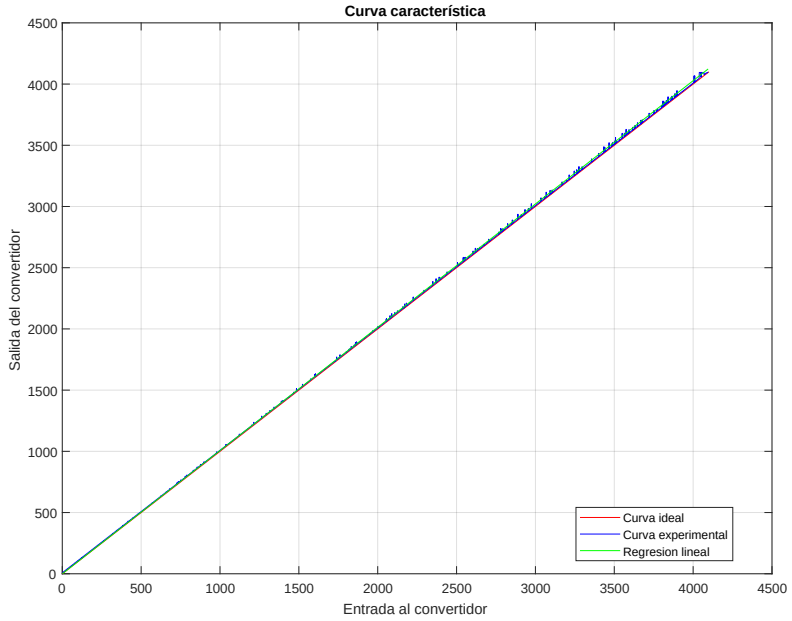

Figura 5.4 Curva característica del CAD.

Se representa en rojo, la curva característica de un convertidor ideal de 12 bits desde $0V$ hasta $4{,}095V$; para poder comparar con otros parámetros más adelante, se dibuja en verde la regresión lineal de los valores experimentales, haciendo uso de la función **polyfit(x, y, 1);**, donde los dos primeros argumentos son los valores experimentales en los ejes "X" e "Y", y el tercer argumento el orden de dicha regresión lineal a través del análisis de los mínimos cuadrados; de la cual se consigue la pendiente y el corte con el eje de ordenadas. Con esta información más adelante se observarán errores de ganancia y offset con respecto a la curva ideal.
Por otro lado, las medias realizadas en cada tramo de la curva característica fueron siendo redondeadas hasta hacerlas coincidir con valores enteros -como se espera de una función de transferencia de un

convertidor-; además destacar que, en lineas generales, la dispersión en la medida fue incrementándose conforme mayor era el valor de entrada.

- **Error de Offset**: este parámetro se puede observar como la diferencia entre los valores de entrada y salida, al tener una tensión de $0V$ en la entrada. Debido a que se tiene la curva característica en ambos ejes como valores discretos, se mide al comienzo dicha diferencia y se obtiene un error de $+6LSB$.

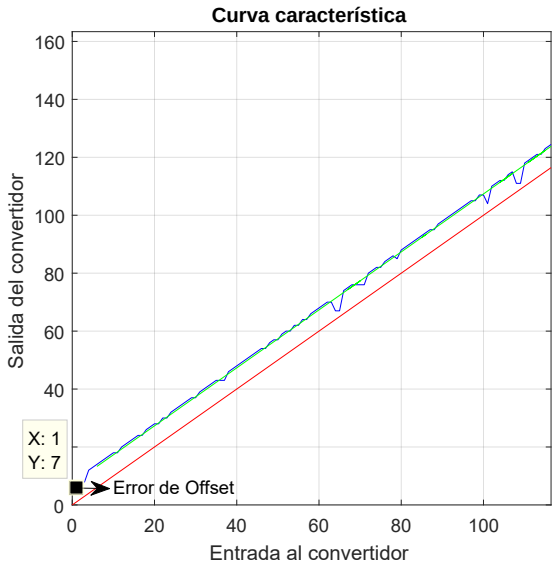

Figura 5.5 Detalle del error en offset.

- **Error de ganancia**: como ocurre con el error de offset, este valor se puede medir de manera inmediata en la curva característica, gracias a la regresión lineal sobre los datos de salida del convertidor. Así pues, viendo la diferencia en el eje vertical entre la curva experimental y la real, se observa un error de $+14LSB$.

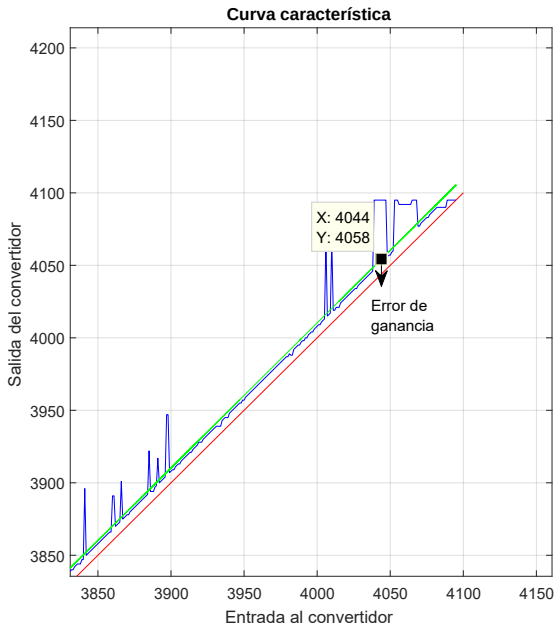

Figura 5.6 Detalle del error de ganancia.

Este punto se comprobó con un multímetro modelo UT33C, para poder asegurarnos de alguna manera que los errores en ganancia no eran debido -al menos en su mayoría- a errores en el convertidor digital-analógico: se escogieron algunos valores al azar en las zonas más altas de la curva característica, se tomaron muestras con el multímetro y, con la salida constante del CDA, se realizó el cálculo de la salida en voltios con respecto a la referencia que se le daba desde el Arduino, y los valores eran coincidentes hasta las décimas de voltio; en conclusión, a partir de 10LSB, podemos asegurar -con los instrumentos que han sido partícipes de los experimentos- que el error en la medida es mayoritariamente independiente del CDA que se usa como generador de señales DC -en estas pruebas-.

- **No linealidad diferencial**: tal y como se había definido anteriormente, la DNL se mide como la proporción en el ancho de la curva característica con respecto al ideal. Debido a que el convertidor digital-analógico usado posee la misma resolución de 12 bits, no se tiene una rampa de tensión como tal -o al menos con mayor resolución que 12 bits-, sino que se compone de pequeños escalones, que de manera teórica son los mismos que pueden observar en el CAD, es por ello además, que tampoco se obtendrán errores no enteros, pues como mínimo el error -si lo hay- es de $1LSB$. Por tener el mismo bit menos significativo, la DNL se puede calcular de la siguiente manera: aquellos valores de entrada que no han tenido una variación en la salida, en prácticamente en todas las ocasiones, el salto era de 1 LSB, así que estos valores poseen un error DNL de $-1LSB$. Por otro lado, los valores que se han leído correctamente por parte del CAD, no tienen ningún error y por último, a partir de 2 repeticiones consecutivas, el error de DNL se suma $+1LSB$.

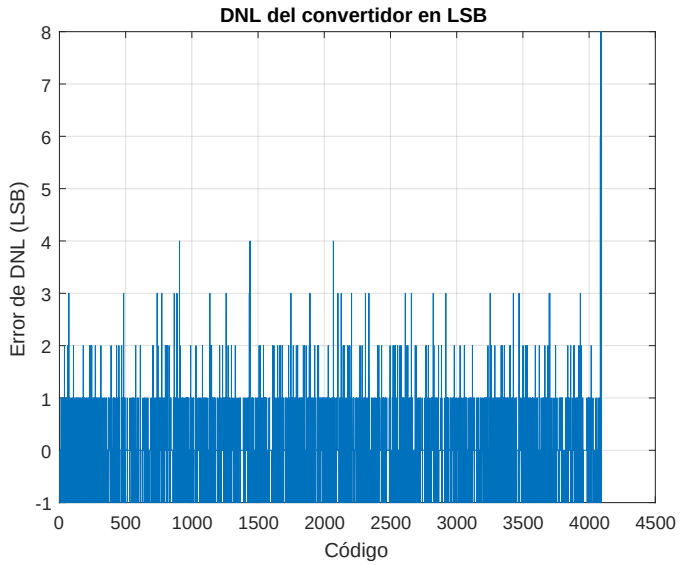

Figura 5.7 No linealidad diferencial experimental.

Así que observando en la gráfica anterior, se obtiene que el mayor error de DNL, se encuentra al final de la curva característica, con un valor de $+8LSB$

- **No linealidad integral**: a raíz de los valores de la no linealidad diferencial, se ha hecho uso de la fórmula teórica detallada en el apartado 2, donde para cada valor de INL_i se van sumando todos los valores DNLs desde 1 hasta i.

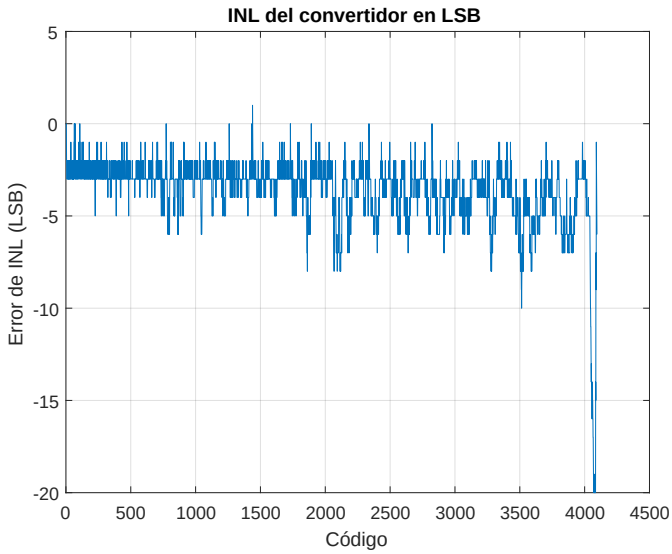

Figura 5.8 No linealidad integral experimental.

El error máximo cometido en la INL es de $-20LSB$ hacia el final de la curva característica, donde la mayoría de los parámetros presentaban mayor dispersión en las medidas.

5.2.3 Parámetros dinámicos del convertidor

Para poder realizar las pruebas dinámicas, se creó una senoide discretizada de 4000 valores en MATLAB con el siguiente algoritmo:

```
t = [0:1:4000];
fs = 4000;
Input1 = sin((2*pi*t)/fs) + 1;
quant=max(Input1)/(3354); %Debido a que el CDA va de 0V a 5V
y=round(Input1/quant);
```

Código 5.1 Creación de la senoide discreta.

Destacar que la cuantización del seno se hizo de 0 a 3354, pues dicho valor es el correspondiente a $4{,}095V$, según una función de interpretación lineal del convertidor digital-analógico. De aquí, se extrajo un cuarto de período, pues con el y algunas sencillas operaciones se puede conseguir el seno completo sin desbordar la memoria del microcontrolador Arduino que está gobernando al CDA. De esta manera se consiguió un período completo de 4000 valores distintos entre $0V$ y $4{,}095V$ con una buena resolución; a costa de poseer una baja frecuencia en la señal -$2Hz$ aproximadamente-, debido a que se priorizó conseguir una señal lo más "analógica" posible frente a la rapidez de la misma; inclusive se modificaron parámetros de las librerías de Arduino, como la frecuencia de la comunicación I2C a unos $700KHz$. El código generador de periodos de senoides es el siguiente:

```
for (counter = 0; counter < 1000; counter ++){
        dac.setVoltage(seno[counter], false);
    }
    for (counter = 1000; counter > 0; counter --){
        dac.setVoltage(seno[counter], false);
    }
    for (counter = 0; counter < 1000; counter ++){
        dac.setVoltage(3354 - seno[counter], false);
    }
    for (counter = 1000; counter > 0; counter --){
        dac.setVoltage(3354 - seno[counter], false);
    }
```

Código 5.2 Generador de senoide discreta.

Por otro lado, se añadió un filtro RC paso baja con una frecuencia de corte de $7{,}234Hz$, con el fin de eliminar los cambios bruscos en la señal de entrada al convertidor analógico-digital, generando ruido de frecuencias más altas a las de estudio. Este fue el circuto con los valores de resistencia y capacidad:

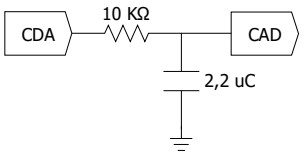

Figura 5.9 Circuito de filtrado paso baja.

- **Relación señal/ruido**: para el cálculo del SNR se ha hecho uso de las funciones nativas existentes en MATLAB; sin embargo, se comenzará por observar el espectro en frecuencia de la señal entrada al CAD:

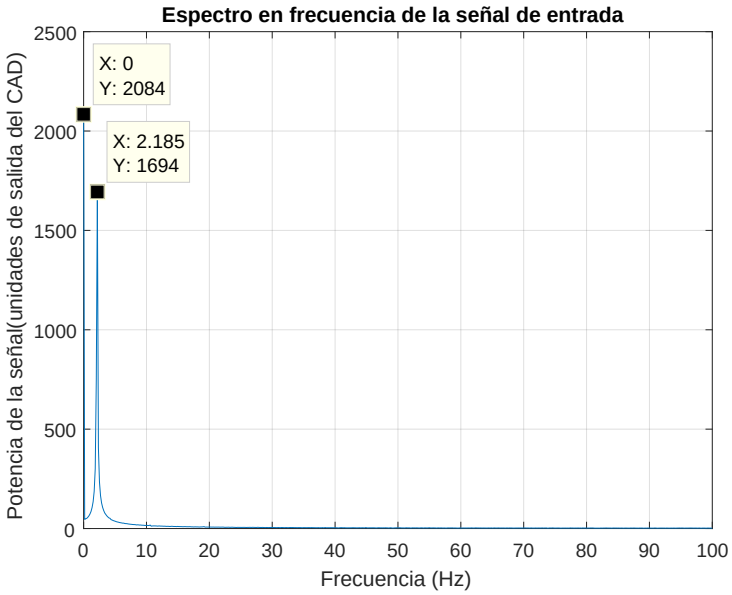

Figura 5.10 Espectro en frencuencia.

De la gráfica, se puede extraer que en la frecuencia $0Hz$ se encuentra el offset de la señal de entrada de aproximadamente la mitad de la amplitud -pues se recuerda que el convertidor que se está estudiando tiene entrada unipolar- y la señal creada en el CDA de un poco más de $2Hz$, frecuencia que viene dada de la propia programación del CDA con el Arduino; se comparó además con funciones orientadas a calcular el tiempo desde que comenzó el programa y los resultados concuerdan perfectamente, un período de senoide completa se encontraba en el entorno de los $478ms$ que en frecuencia sería $2{,}09Hz$. Además se observa una buena filtración en frecuencias inmediatamente más altas, que nos permite esperar unos resultados más independientes de la señal de entrada.

Con respecto al SNR, con la función **snr(x,Fs)** de **MATLAB**, se calcula el valor en dB, sus primeros armónicos y el valor de DC de la señal. Se recuerda además que la frecuencia del convertidor **Fs** es de $200Hz$

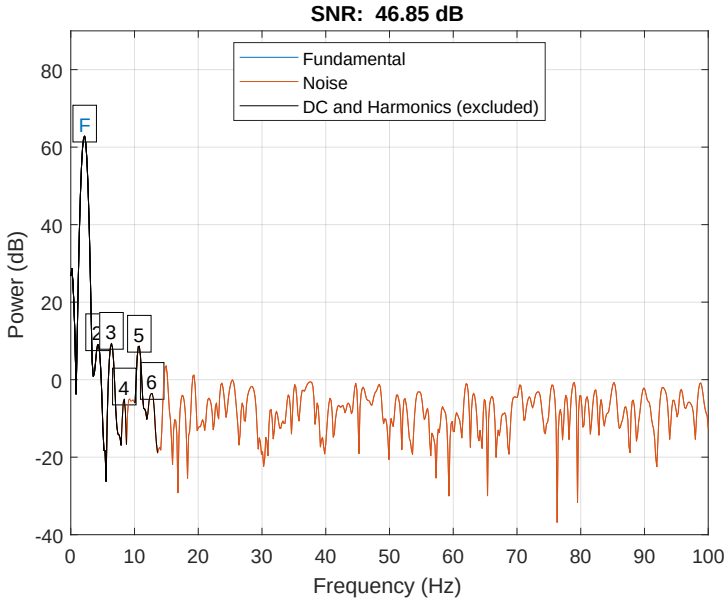

Figura 5.11 Relación señal/ruido.

- **ENOB**: para el número efectivo de bits no se realiza ninguna medición directa sobre el convertidor, lo que se hace es el cálculo teórico del apartado 2 tras haber obtenido el valor de la relación señal/ruido, de la siguiente manera:

$$ENOB = \frac{SNR - 1{,}76}{6{,}02} \rightarrow ENOB = \frac{46{,}85 - 1{,}76}{6{,}02}$$

- **Distorsión armónica**: como ocurre con el apartado anterior, **MATLAB** también posee una función para éste cálculo: **thd(x,Fs)**, la cual por defecto, contará con los primeros 6 armónicos.

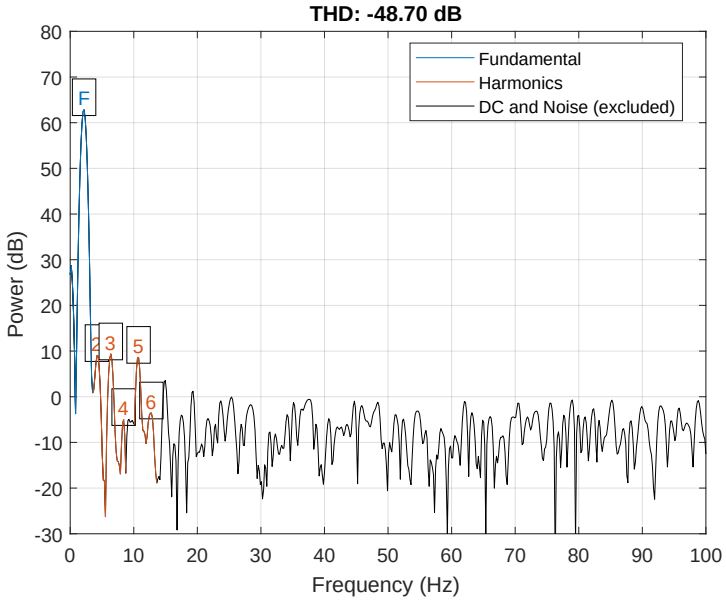

Figura 5.12 Distorsión Armónica Total.

Explicado al principio, estas medidas se realizaron con un filtro RC paso baja a la entrada del CAD para obtener resultados más independientes del CDA. Ahora se procede a ver los mismos resultados pero sin dicho filtro, para poder tener una mejor perspectiva de la mejoría en los resultados y su influencia más adelante tanto, en la relación señal/ruido como en los armónicos.

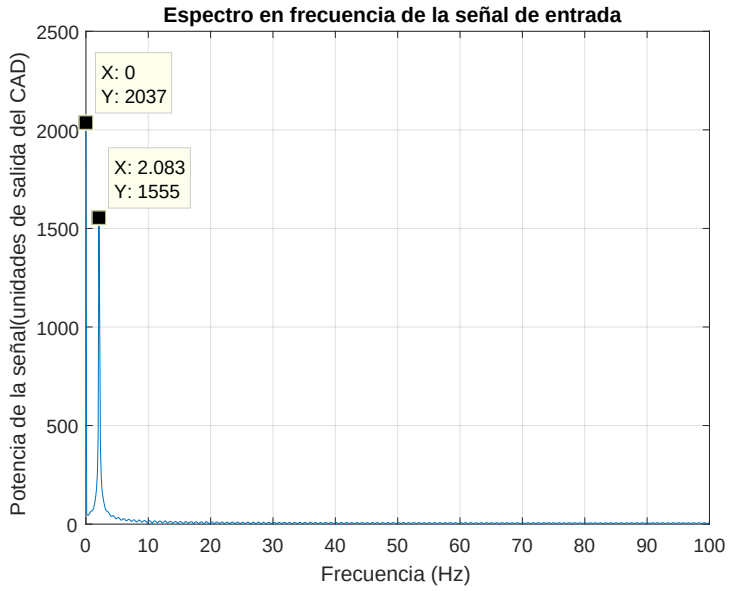

Figura 5.13 Espectro en frecuencia sin filtrado a la entrada.

Se acompaña además, un detalle de las frecuencias más altas -hasta $100Hz$- para comprobar que, efectivamente, el filtrado se estaba realizando correctamente.

Por lo general, se espera que los parámetros dinámicos de estudio mejoren conforme los componentes de más alta frecuencia que la de interés tengan una amplitud menor; así que se adjuntan los mismos experimentos que anteriormente sin la posibilidad, a priori, de conocer con exactitud cuánta mejoría existe en la señal de entrada al convertidor analógico-digital.

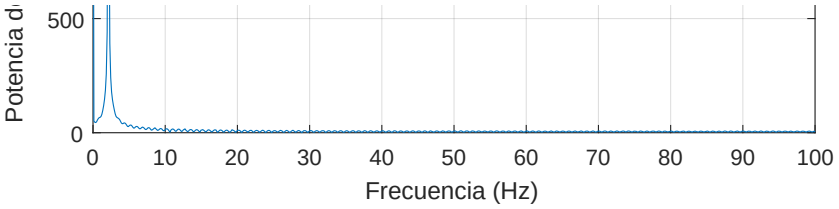

Figura 5.14 Detalle de las frecuencias altas sin filtrar.

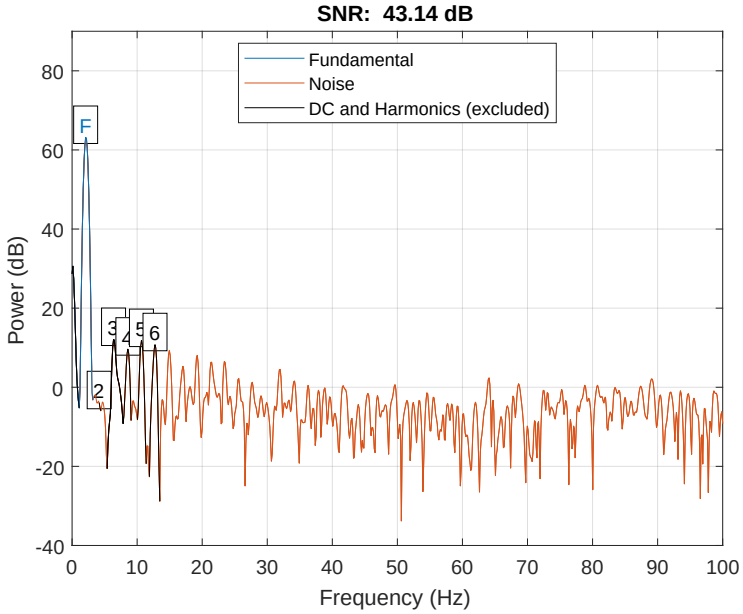

Figura 5.15 Relación señal/ruido sin filtrado a la entrada.

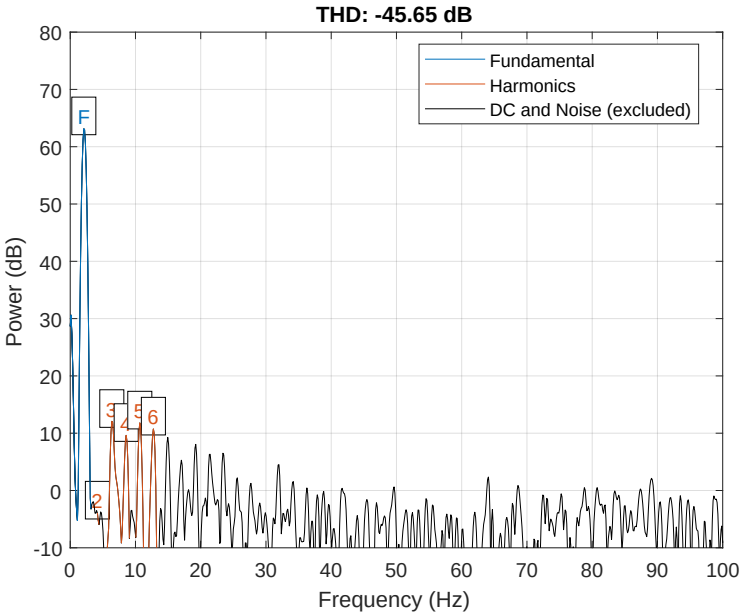

Figura 5.16 Distorsión Armónica Total sin filtrado a la entrada.

Como se puede comprobar, tanto en la relación señal/ruido como en la distorsión armónica, se obtiene una mejora de unos $3dB$ con este sencillo filtrado; lo cual a pesar de estar lejos de las medidas deseables -como se detallará en el siguiente punto-, da pie a pensar que introduciendo pequeñas mejoras en el circuito de pruebas, se puede esperar un avance de carácter importante.

6 Resultados y conclusiones

La mayoria de las ideas fundamentales de la ciencia son esencialmente sencillas y, por regla general, pueden ser expresadas en un lenguaje comprensible para todos.

ALBERT EINSTEIN

Tras haber visto las distintas pruebas y experimentos realizados al sistema y al convertidor de manera más concreta, se procederá al análisis de los resultados, su extrapolación para una mayor cantidad de canales y algunas conclusiones finales.
Con respecto a las restricciones temporales de conversión de datos tenemos que, los tiempos consumidos en un convertidor permiten un buen margen para meter otros dos convertidores de las mismas características y leer 24 canales. En el siguiente diagrama temporal se explica más detalladamente

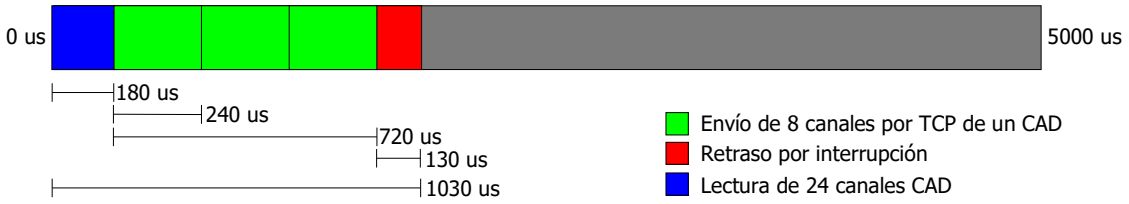

Figura 6.1 Ciclo de lectura de los datos del convertidor.

La ejecución del programa, como se dijo anteriormente, es cada período de la frecuencia de Nyquist, es decir: $f = 200Hz \rightarrow T = 1/200seg = 0{,}005seg = 5000us$; durante ese tiempo se deben realizar todas las operaciones oportunas, que efectivamente, dan lugar sin problemas. En el diagrama anterior, se observa que en la zona azul, se ha incluido el tiempo que tardaría en leerse 24 canales (3 convertidores, con una media de $60us$ por cada uno), en las tres zonas verdes, el tiempo de envío TCP medio por cada CAD, siendo la suma de tres un total de $720us$ y en último lugar, el tiempo de $130us$ que algún canal ha tardado en enviar su trama TCP; la explicación es que durante el envío de alguna trama, puede saltar la interrupción para la lectura, la cual tarda del orden de $60us$, sin embargo, el cambio en las lineas de ejecución, para saltar a una función distinta y volver donde estaba, añade el tiempo restante a dicho retraso en el envío la trama. En la zona gris por tanto, radica el tiempo que el microcontrolador se queda "libre" realizando una espera activa, a pesar de ello, se llevó a cabo la programación de colas *FIFOs* por prevención.
En lo que se refiere a los parámetros calculados de manera experimental del convertidor analógico-digital dentro del sistema, desde una perspectiva general, no son los adecuados que debiera tener un CAD de 12 bits; además las mediciones se realizaron sobre un sólo canal, aunque el envío desde el *Feather Huzzah* era de los 8 al completo; con lo que, en cuestiones de retraso en las medidas por los cambios de canal, se siguen viendo reflejados en los experimentos realizados. Así que procede a algunos apuntes: con respecto a la calidad de los experimentos debido al circuito, condiciones y dispositivos que han intervenido en las pruebas; se puede entender una desmejoría comparando con los valores de fábricas provenientes del *datasheet*:

Tabla 6.1 Parámetros del fabricante [12] y experimentales.

Parámetro	Fabricante	Experimental
SNR	$70dB$	$46{,}85dB$
THD	$-77dB$	$-48{,}70dB$
Resolución	12 bits	7 bits
DNL	$-0{,}9/1{,}5$ LSB max	$+8$ LSB max
INL	± 1 LSB max	-20 LSB max
Error en ganancia	$\pm 1{,}5$ LSB max	$+14$ LSB
Error de offset	± 8 LSB max	$+7$ LSB

Con respecto a la relación señal/ruido, en primer lugar detallar que, el valor de este parámetro dado por el fabricante son referidos a con amplitud en la señal de entrada $5V$, mientras que nuestra referencia se encuentra en $4{,}095V$, una diferencia en la potencia en la señal que permitiría tener un mayor SNR.
Otras posibles fuentes de error en las medidas han podido ser: el ruido generado por el circuito implementado en protoboard, pues se ha usado cables *dupont* para prototipado en el conexionado y esto genera incertidumbre en la medida. Por otro lado, ruido generado en la alimentación al sistema, pues se ha hecho uso de la alimentación USB de $5V$ generada por un ordenador portátil, para proveer al microcontrolador y este con su referencia de $5V$ alimentar al convertidor analógico-digital. Sin embargo, para descartar posibles fuentes de ruido, se probó a separar la fuente de tensión de cualquier otro dispositivo electrónico; sustituyendo la alimentación USB por una batería Li-Po de $3{,}7V$ adecuada para el *Feather Huzzah*, para intentar evitar posibles errores en la referencia del convertidor, pero no se mostró ningún cambio en los valores de salida de los mismos experimentos.
Todos estos orígenes de incertidumbre en la medida se extrapola también para los demás parámetros, pues pequeños detalles van mermando los valores como la distorsión armónica; aunque en este caso, no parece que aumentar la resolución de la señal de entrada vaya a incrementar de manera brusca la calidad de estos parámetros, si no que dependen de los otros puntos ya anteriormente descritos. Sobre el número de bits efectivos, detallar que se encuentra cerca de ser un convertidor de 8 bits, lo que es una diferencia importante en varios ámbitos, desde que se convierta en un convertidor -o sistema- no válido, como pérdida económica por tener un CAD de 12 bits trabajando con menos, una cuestión que no puede resultar aceptable bajo ningún concepto.
En general, los parámetros estáticos tampoco son los idóneos, pues en la mayoría de los valores descritos, existen problemas sobre todo en la zona de tensiones más altas. Por ejemplo, el error de offset se encuentra en valores semejantes a los dados por el fabricante, en cambio, el error de ganancia se aleja bastante más en proporción, debido a que se mide en las zonas más altas pues normalmente la pendiente de la curva experimental es algo mayor que la pendiente de la curva característica ideal. Con respecto a los parámetros de no linealidad, se constata el mismo hecho, en las zonas de mayor tensión surgen mayores problemas; aunque en las demás zonas, tampoco resultan unos buenos valores con respecto a los del fabricante; todo esto conlleva una pérdida de información que se puede ver reflejada en el número de bits efectivos, pues se genera el efecto que ya se explicó de los *missing codes*, donde el convertidor no ha sido capaz de discernir algunos cambios en la entrada.

6.0.1 Conclusiones

- Con respecto al resultado de las pruebas y experimentos; decir que, por lo general, no se pueden considerar como concluyentes debido a la resolución y fiabilidad de los equipos que han intervenido en nuestro banco de pruebas. Sin embargo, es cierto que analizando y comparando como se hizo por ejemplo con el multímetro, difícilmente se van a obtener los mismos valores que los dados por el fabricante, pues antes las mismas condiciones, se obtenían errores mayores a $10LSB$ comparando el CAD con el multímetro.
  A pesar de ello, si alguien sigue por esta línea de trabajo, se anima a seguir estos pasos en laboratorio para obtener rápidamente resultados y analizar si el sistema al completo no se puede considerar válido o el convertidor difiere bastante de los parámetros dados por el fabricante.
- Antes de volver a hacer las pruebas, algunos detalles que potencialmente deben hacer mejorar el resultado de las mismas: diseño de un circuito impreso PCB para minimizar el ruido en los contactos entre los cables y las placas de desarrollo, hacer uso de una alimentación estable y con bajo ruido para la referencia del convertidor; también acceder a un generador de señales que esté calibrado y con una

resolución mayor que el CDA utilizado. Por último, cambiar la referencia de tensión del convertidor que tiene fijada a $4{,}095V$, que efectivamente nos permite tener números más sencillos de manipular, pero es otro punto añadido que nos aleja de los parámetros del *datasheet*.

- A pesar de todo ello, no se puede asegurar con certeza que en otras circunstancias donde el acceso a equipos de laboratorios si hubiera sido posible, los resultados habrían sido mucho mejores; aunque potencialmente se entiende que sí. Sin embargo y ateniéndonos a las pruebas se han presentado, el sistema actualmente no se puede considerar como válido para la recopilación de señales de electroencefalografía.

7 ANEXO

```
#include "espressif/esp_common.h"
#include "esp/uart.h"
#include "FreeRTOS.h"
#include "task.h"
#include "esp8266.h"
#include <stdio.h>
#include "esp/spi.h"
#include "queue.h"
#include "ssid_config.h"
#include <string.h>
#include <lwip/api.h>

#define TELNET_PORT 23

const int CS = 0;
static volatile uint16_t CADrx;
const int freq = 200; //Hercios
static QueueHandle_t CADcola;

void lectura_interrupt_handler(void *arg)
{
  uint16_t aux;

  gpio_write(CS, 0);
  aux = spi_transfer_16(1,0x8310); //canal 0
  gpio_write(CS, 1);
  CADrx = aux & 0x7FFF;
  xQueueSendToBackFromISR(CADcola, &CADrx, NULL);

  gpio_write(CS, 0);
  aux = spi_transfer_16(1,0x8710); //canal 1
  gpio_write(CS, 1);
  CADrx = aux & 0x7FFF;
  xQueueSendToBackFromISR(CADcola, &CADrx, NULL);

  gpio_write(CS, 0);
  aux = spi_transfer_16(1,0x8B10); //canal 2
  gpio_write(CS, 1);
  CADrx = aux & 0x7FFF;
  xQueueSendToBackFromISR(CADcola, &CADrx, NULL);

  gpio_write(CS, 0);
  aux = spi_transfer_16(1,0x8F10); //canal 3
  gpio_write(CS, 1);
  CADrx = aux & 0x7FFF;
  xQueueSendToBackFromISR(CADcola, &CADrx, NULL);

  gpio_write(CS, 0);
  aux = spi_transfer_16(1,0x9310); //canal 4
  gpio_write(CS, 1);
  CADrx = aux & 0x7FFF;
  xQueueSendToBackFromISR(CADcola, &CADrx, NULL);
```

```
53
54    gpio_write(CS, 0);
55    aux = spi_transfer_16(1,0x9710); //canal 5
56    gpio_write(CS, 1);
57    CADrx = aux & 0x7FFF;
58    xQueueSendToBackFromISR(CADcola, &CADrx, NULL);
59
60    gpio_write(CS, 0);
61    aux = spi_transfer_16(1,0x9B10); //canal 6
62    gpio_write(CS, 1);
63    CADrx = aux & 0x7FFF;
64    xQueueSendToBackFromISR(CADcola, &CADrx, NULL);
65
66    gpio_write(CS, 0);
67    aux = spi_transfer_16(1,0x9F10); //canal 7
68    gpio_write(CS, 1);
69    CADrx = aux & 0x7FFF;
70    xQueueSendToBackFromISR(CADcola, &CADrx, NULL);
71
72  }
73
74
75  void envio_CAD(void *pvParameters)
76  {
77
78    struct netconn *nc = netconn_new(NETCONN_TCP); //Creamos un nuevo identificador de conexion
79
80      netconn_bind(nc, IP_ANY_TYPE, TELNET_PORT); //Enlazamos a traves del puerto 23, cualquier IP
81      netconn_listen(nc); //Le decimos a la conexion que se ponga en modo escucha
82    struct netconn *client = NULL;
83          err_t err = netconn_accept(nc, &client); //Aceptamos nueva conexion
84    err_t err_c = ERR_OK;
85    uint16_t CAD;
86    char buf[6];
87
88    while(1) {
89
90      if (err_c == ERR_OK & err == ERR_OK){
91      QueueHandle_t *CADcola = (QueueHandle_t *)pvParameters;
92      if(xQueueReceive(*CADcola, &CAD, 1000)) {
93        snprintf(buf, sizeof(buf), "%u", CAD);
94        err_c = netconn_write(client, buf, sizeof(buf), NETCONN_COPY);
95        char buf[80];
96            snprintf(buf, sizeof(buf), "%u", CAD);
97          netconn_write(client, buf, strlen(buf), NETCONN_COPY);
98        }
99      }
100     else {err = netconn_accept(nc, &client);
101       if (err == ERR_OK) {err_c = ERR_OK;}}
102   }
103 }
104
105
106
107 void user_init(void)
108 {
109   uart_set_baud(0, 115200);
110   sdk_system_update_cpu_freq(SYS_CPU_160MHZ) ;
111   spi_init(1, SPI_MODE0, SPI_FREQ_DIV_20M, 1, SPI_BIG_ENDIAN, true); // inicialización de SPI
112
113     ////////////////////// Envio de la configuracion inicial del SPI ///////////////////////
114   gpio_enable(CS, GPIO_OUTPUT);
115   gpio_enable(2, GPIO_OUTPUT);
116   gpio_write(2,0);
117   gpio_write(CS, 1);
118   sdk_os_delay_us(60000); //uint16_t us
119   sdk_os_delay_us(40000);
120   gpio_write(CS, 0);
121   CADrx = spi_transfer_16(1, 0x8310);
122   gpio_write(CS, 1);
123
```

```
124     ////////////////////// Creacion y configuracion de las interrupciones ///////////////////
125
126   /* Paramos ambos timers y enmascaramos como precaucion */
127   timer_set_interrupts(FRC1, false);
128   timer_set_run(FRC1, false);
129
130   /* Configuramos el ISRs */
131   _xt_isr_attach(INUM_TIMER_FRC1, lectura_interrupt_handler, NULL);
132
133   /* Configuramos la frecuencia */
134   timer_set_frequency(FRC1, freq);
135
136   /* Desenmascaramos y echamos a andar los timers */
137   timer_set_interrupts(FRC1, true);
138   timer_set_run(FRC1, true);
139
140     ////////////////////////////////////////////////////////////////////////////////////////
141
142     ////////////////////// Configuración de conexion WiFi ///////////////////////////////////
143
144       struct sdk_station_config config = {
145           .ssid = "Livebox-E202",
146           .password = "4MN78K78",
147       };
148
149     /* required to call wifi_set_opmode before station_set_config */
150       sdk_wifi_set_opmode(STATION_MODE);
151       sdk_wifi_station_set_config(&config);
152       sdk_wifi_station_connect();
153
154     ////////////////////////////////////////////////////////////////////////////////////////
155
156   CADcola = xQueueCreate(20, sizeof(uint16_t));
157   xTaskCreate(envio_CAD, "envio_CAD", 2048, &CADcola, 2, NULL);
158 }
```

Código 7.1 Código del microcontrolador *Feather Huzzah*.

Índice de Figuras

Índice de Tablas

Índice de Códigos

Bibliografía

[1] Mayo Foundation for Medical Education and Research. Electroencefalografía (EEG) - Mayo Clinic.

[2] Espressif Systems. Espressif Smart Connectivity Platform: Esp8266. *WiFi Allience*, page 23, 2013.

[3] MicroChip. 12-Bit Digital-to-Analog Converter with EEPROM Memory in SOT-23-6. *Microchip Technology Inc*, pages 1–42, 2007.

[4] SuperHouse Automation. GitHub - SuperHouse/esp-open-rtos: Open source FreeRTOS-based ESP8266 software framework.

[5] Espressif. GitHub - espressif/esptool: ESP8266 and ESP32 serial bootloader utility.

[6] Brian Black. Analog-to-Digital Converter Architec-tures and Choices for System Design. *Analog Dialogue*, page 1, 1999.

[7] NYUST MSIC D&T Lab., Dept. of El. Eng. DAC & ADC Testing Fundamental Outline Specifications of DAC Specifications of ADC Test methodology. page 75, 2009.

[8] Embedded Webpage. Understanding analog to digital converter specifications - Embedded.com, 2005.

[9] Wikipedia. Ruido de cuantificación - Wikipedia, la enciclopedia libre, 2019.

[10] Alfredo Pérez Vega-Leal. 7. Ruido en CAD y CDA. pages 1–32, 2017.

[11] Luis Llamas. El bus SPI en Arduino, 2016.

[12] Analog_Devices. 8-Channel, 1 MSPS, 8-/10-/12-Bit ADCs with Sequencer in 20-Lead TSSOP. *Analog Devices*, pages 1–28, 2014.

[13] James F Kurose. *Redes de Computadoras. Un enfoque descendente*.